高速网络中端到端 QoS 传输调度算法的研究

Research on the End-to-End QoS Scheduling Algorithm of High Speed Networks

马 涛 石冰心 曾庆徽 倪 强

(华中理工大学电子与信息工程系 武汉430074)

Abstract This is a review paper on the end-to-end quality of service(QoS) scheduling algorithm of high speed networks. In this paper, we analyze the various alogrithms that have been proposed, and disscuss some of the trade-offs they involve. A few of issues, research directions and open problems in this area are discussed.

Keywords QoS, Scheduling algorithm, Queue

1 前言

以 Internet 为代表的信息产业标志着人类社会进入了知识经济时代,信息的获取、传送、存储和处理之间的孤岛现象随着计算机网络和多媒体技术的发展而逐渐消失。IP 电话、视频会议、电子商务等越来越多的网络应用涌现出来,这使得 IPv4本身的缺点日益变得明显起来:没有带宽控制和流量控制功能,服务质量 QoS(Quality of Service)没有保证,忙时分组丢失非常严重,极大地影响了网络传输的效率。其中最为突出的是 IPv4对于时间要求颇高的数据包(如视频、音频数据包)和一般性数据包(如文件传送、电子邮件等数据包)的处理并不加以区分。对于连贯性和时间性要求很高的视频和音频数据来说,数据丢失将是一个致命的问题。

端到端 QoS 可以保证数据包不仅能到达其欲传输的目的地址,而且可以保证数据包的顺序性、完整性和实时性。传输调度算法是端到端 QoS 的核心技术之一。端到端 QoS 传输调度算法通过对数据包进行合理的排队,对含有内容标识的数据包进行优化,并对其中特定的数据包赋以较高的优先级,从而加速传输进程,并实现实时交互。

2 端到端 QoS 传输调度算法

基于端到端 QoS 传输调度算法的基本功能是从节点的每一个输出链路中挑选在下一个有效周期发送的分组,并决定其发送顺序和带宽占用。QoS 传输调度要基于几个原则:带宽的保证、流的隔离、时延的保证和公平选择等。根据其采用的调度算法和网络链接上复用的会话特性,可建立相应网络模型,并计算出一定的网络性能和端到端 QoS 保证。协议和算法的复杂性要适应网络高速传输和便于实现,使其具有可扩展性和鲁棒性。在选择一种调度算法时主要从几方面进行考察:

* 调度算法在本质上提供不同类型服务性能上的保证?例如:能使网络提供基于每一会话的速率保证,或是否以一种简单的优先级结构,可以保证一种会话比另一种会话有高的优先级别。

* 算法在提供服务保证上的效率。基本上来说,同时有多少给定特性的会话可以在链路上复用。这通常可以用呼叫接纳和可调度区域来表示。

* 算法复杂性,就今天的千兆以太网、ATM 等高速网络而言,在短暂的时间间隔内,即可造成大量数据包在缓存中的排队等待,调度单一数据包的时间在持续下降。因此算法应可能简单,减少对每一数据包的调度时间。

* 进行调度决策时需采用的参数和算法。目前比较特别感兴趣的算法是否能提供基于会话的时延或速率保证[1,2]或两者的结合[3~5],单独意义上的速率或时延保证可使得算法有更大的灵活性。

* 当所处理的流量超过了服务保证需求的弹性,有些算法可以轻松处理过载的流量,而有些算法需要一些额外的机制来保障。算法在处理过量会话时,如何将当前有效带宽在不同会话中进行公平分配,

网络用于处理过量的输入流量的方法就是采用队列调度算法来对要服务的流量进行排队,然后决定在

马 涛 博士,主要研究领域为高速网的 QoS,石冰心 教授,博士生导师,近来的研究领域是现代信息网络的关键技术及其应用。

输出链路上的优先顺序。下面分析一下目前已存在的各种调度算法。

2.1 FCFS(先来先服务)

这是一种最简单的调度算法，数据包以其到达顺序而接受调度服务。FCFS 不需要额外配置、易于实现，不需要维护基于每一会话的状态，特别是，从等待队列中插入或删除数据包的时间开销为常量。FCFS 算法作为一种基本的调度算法，运行在各种交换机和路由器中，但 FCFS 算法不能提供很好的时延或速率保证，是通过限制等待队列长度来实现的。

FCFS 缺少对资源的公平分配和共享，可以通过与相应缓冲区管理方案相结合来提供一种有效的带宽分配。

2.2 FPS(固定优先级调度)

这种方式能以一种相当粗糙的方式提供不同级别服务保证的能力(如图1)。会话根据固定的静态数目优先等级进行分类，对不同的优先级维护一个不同的队列。链路复用器对数据包调度只有在所有高优先级队列为空时才调度低调先级队列。在相同优先级队列中数据包以 FCFS 方式被调用，这种算法的复杂性在于提供对几种不同优先级队列的维护，调度一个数据包的时间开销为常量，仅依赖于优先级别数目，而独立于当前链路上会话数。

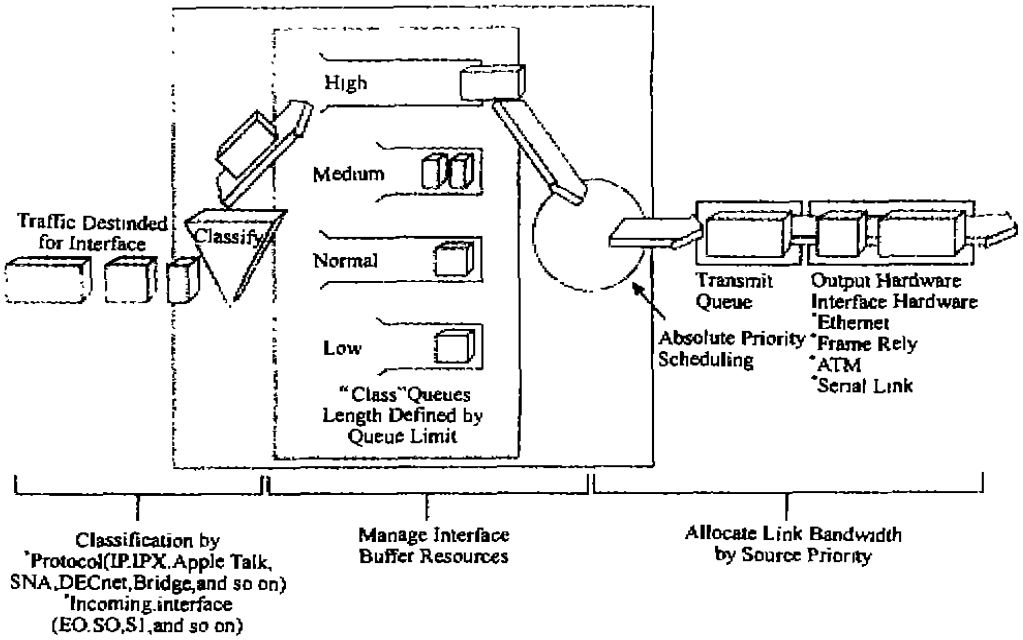

图1 优先级队列将数据包放入相应的队列并调度

优先级调度提供了一定意义上的服务质量和基于每一个会话的端到端 QoS 保证。然而，此算法只能保证在一条链路上一类会话能得到比另一类会话更好的服务。同 FCFS 一样，通过对队列长度的限制来保证端到端时延。然而由于低优先级数据包只有在所有高优先级数据包传输之后才能接受服务，这势必会影响低优先级数据包的时延差。总的来说，FCFS 和 FPS 在提供端到端时延上并不理想。

另一个问题是不同的交换机可能有不同数量的优先级，而对优先级在不同交换机上的映射使得端到端 QoS 难以预测。

2.3 WFQ(加权式公平队列)

WFQ 调度算法的研究目前比较流行，并有许多相应衍生算法。WFQ 克服了 FCFS 和 FPQ 调度算法的缺点，对接收到的会话有较好控制，这允许网络提供基于每一会话的端到端时延保证。WFQ 根据会话在带宽中所占的比例来以一种公平方式对待过量的流量(如图2)[6]。

大多数的 WFQ 衍生算法，是基于 GPS(广义处理机共享)[7,8]的调度算法。在 GPS 调度中，定义了一个流体的会话模型，这样会话在通过许多链路时，且途中交换机均采用 GPS 的调度算法时，端到端时延便能够计算出来。GPS 的思路是：一个加权值 φ_i 和一个相应会话 i 相联系，$i=1,\cdots,N$，并且链路容量在共享链路上的带宽分配直接与当前活动的会话加权值成比例。换言之，若以 γ 表示链路带宽，那么会话 i 可通过获得一个最小服务速率为 $(\varphi_i/\sum_{i=1}^{N}\varphi_i)\gamma$ 来保证服务质量。若在任何给定时间间隔内会话 i 用不完分配给自己的

带宽，即有多余的带宽，可被其他活动会话来共享。GPS 的调度算法根据当前活动的会话，以加权值成比例地共享多余带宽。若以 B(t) 表示时刻 t(t≥0) 时活动的会话集合，那么会话 i 的速率保证是通过接受一个最小速率 r_i 来保证：

$$r_i(t)=\begin{cases}(\varphi_i/\sum_{j\in B(t)}\varphi_j)\gamma, & i\in B(t)\\ 0, & \text{其它}\end{cases}$$

GPS 的优点在于公平处理过量的数据流量。若某一时间间隔内链路上仅有会话 i，j，则它们相应所接受到的链路带宽直接与其加权值成正比，而无论是哪一个会话有过量流量。以 $W_i(t_1,t_2)$ 表示在时间间隔 t_1,t_2 内数据所得到的带宽，$i=1,\cdots,N$，则 GPS 可保证当前仅活动的两个会话所得到的带宽容量为：

$$\frac{W_i(t_1,t_2)}{\varphi_i}=\frac{W_j(t_1,t_2)}{\varphi_j}$$

这种机制保证了 GPS 调度的公平性，相应对任何算法我们都可以用 $|W_i(t_1,t_2)/\varphi_i-W_j(t_1,t_2)/\varphi_j|$ 来比较不同调度算法对任一会话的公平性。在 GPS 中，该值为0，可以体现出很好的相对的公平性。

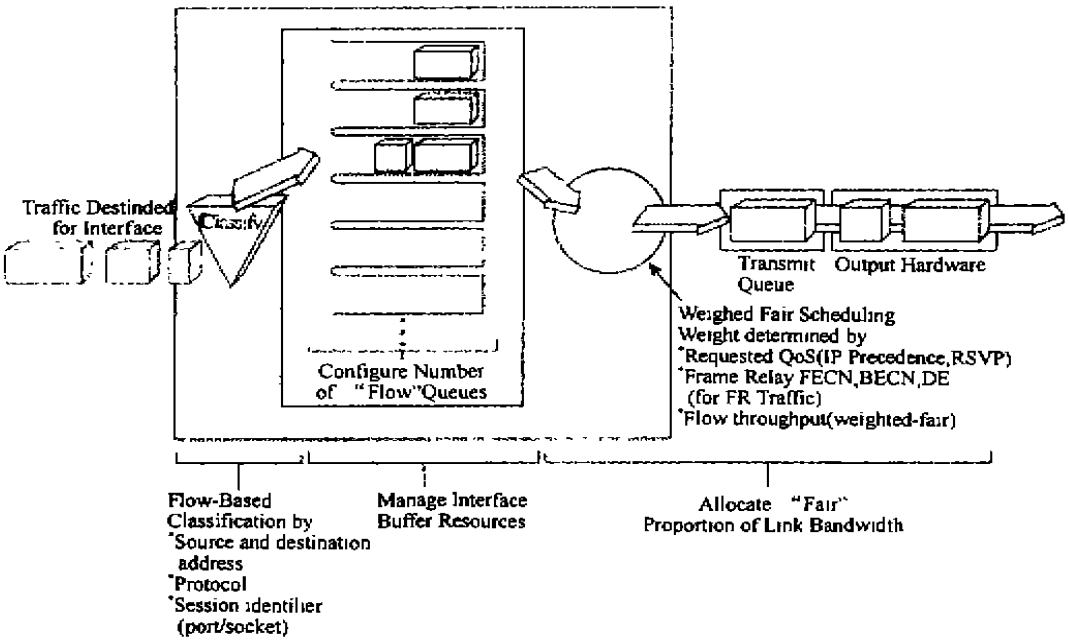

图2　采用 WFQ 算法，若当前有许多活动的高数据量会话，则其传输速率、时延和抖动则可预测

事实上，并没有真正意义上的流体数据流，GPS 仅作为一种参考模型。通过类似这种参考模型定义的相应调度算法 PGPS(打包式广义处理机共享)，又称为 WFQ 算法。PGPS 可认为对 GPS 参考模型的一种在现实中的抽象[9]。

PGPS 调度算法利用了虚拟时钟的概念，主要用以定义时间间隔。虚拟时钟 V(t) 在链路繁忙的起点定义为0，其中 V(t) 以 $\frac{\partial V}{\partial t}=\gamma/\sum_{i\in B(t)}\varphi_i$ 方式递增，基于这个虚拟时钟的概念，可对每一会话中任一数据包进行定义。例如：第 i 个会话中的第 k 个数据包。若设虚拟起始时间为 s_i^k，虚拟结束时间为 f_i^k，则可表示出任一数据包在 GPS 参考系统中的开始时间与结束时间。若进一步表示到达时间为 a_i^k，和它的长度为 l_i^k，可得到如下关系：

$$f_i^0=0,$$
$$s_i^k=\max\{V(a_i^k),f_i^{k-1}\},k=1,\cdots$$
$$f_i^k=s_i^k+l_i^k/\varphi_i,k=1,\cdots$$

PGPS 工作方式是：调度算法总是选择有最小虚拟结束时间的数据包在链路上发送，这可以以一种基于数据包虚拟结束时间为优先级队列来进行传输。从队列中插入和删除的时间复杂度为 O(logN)。然而，在虚拟时钟递进计算中有一定时间开销，因为在整个系统繁忙期间 B(t) 集合可能会变化。在最坏的情况下，虚拟时钟的时间复杂度可能为 O(N)。

PGPS 另一个优点在于端到端的时延范围可以根据每一会话的加权值计算出来。假设共有 H 段链路，以 $\gamma^h(h=1,\cdots,H)$ 表示会话 i 所经过各链路的速率。假设会话 i 的流量特性为 $A_i(t)=b_i+r_i,t,t\geq 0$，以 R_i 表示会话 i 在经过的所有链路上的最小保证速率。假设在网络稳定的条件下，$R_i\geq r_i$，则会话端到端的时延为：

$$\hat{D}_i=b_i/R_i+(H-1)M_i/R_i+\sum_{h=1}^{H}L_{max}^h/\gamma^h$$

此处 M_i 表示会话 i 中最大数据包长，L^h_{max} 表示在链路上允许的最大数据包长，注意无论经过多少条链路，b_i/R_i 的时延在所有链路上只计算了一次。因为 PGPS 调度算法能有效地平滑会话 i 中的突发数据，所以突发数据所造成的时延 b_i/R_i 在随后链路上不会再遇到了，同时注意端到端时延也独立于会话 i 经过的各条链路上所复用的会话数目，这一特性使得 PGPS 算法成为一种较好的提供牢固的端到端有界时延保证的调度算法。

2.4 基于 WFQ 的改进算法

WFQ 算法的一个主要的缺点在于计算虚拟时钟上的复杂性，对此有不少改进和简化算法，仍能保持有界时延保证特性。

SCFQ（自时钟公平队列）[9]是 WFQ 的一种简单替代算法，不再依靠 GPS 参考模型来计算虚拟时间，减少了需要跟踪虚拟时钟的计算量。时间复杂性的降低导致了比 WFQ 算法高的时延区间，在每条经过的链路上又引入了时延 $\sum_{i=1}^{N} M_i/\gamma^h$，其主要缺点在于时延与每一链路上所复用的会话数目相关。

SFQ（起始时间公平队列）[3]是另一种公平队列调度算法。它很相似于 SCFQ，其主要不同点在于调度时总是选择所有会话中有最小虚拟起始时间的数据包在链路上发送，与 WFQ 算法选择最小虚拟结束时间正相反。其端到端时延很近似于 SCFQ，但相对小一些。

WF^2Q（最坏情况公平加权公平队列，Worst-case Fair Weighted Fair Queueing）[10]算法则同时使用 GPS 参考模型中数据包开始时间和结束时间来更接近 GPS，从而达到更优化、平等、公平。WF^2Q 选择一个参考模型中有最小结束时间并且其虚拟开始时间小于当前时间的数据包进行传输，WF^2Q 的时延区间与 WFQ 相同。类似的算法如文[11,12,13]。

2.5 EDF（最小生存时间优先，Earliest deadline first）

EDF 调度算法是一种动态优先级的调度算法，每一个数据包的优先级在其到达时确定，主要根据其到达时间和所属的会话时延保证要求来设定一个生存时间。

EDF 调度时总是选择当前有最小生存时间的数据包来传输，EDF 的动态特性表现在：随着数据包在系统中等待传输时间越长，相应其优先级越大，这可以保证对于有不同迫切时延要求的数据包也能得到更好的服务，在不牺牲同一复用链路上其它会话的时延保证特性的前提下，这要优于传统的静态优先级方案。

对于任何常用到达特性过程的数据包，均对应一个生存时间，EDF 在保证数据最小和最大时延特性上做得很好。象前面所提到的，GPS 基于分配每一会话的权值来确保时延区间，这种权值与链路上保留速率有关，对于一个有小时延保证的会话相对会分配一个大的带宽，使资源的利用率较低，这样会影响一个有严格时延保证特性的低带宽会话。EDF 的一个突出优点就是将对于一个会话的时延保证与吞吐量隔离开。

在 EDF 算法中，比 FCFS 和静态优先级都复杂得多，复杂性在于算法选取一个有最小生存时间的数据包在链路上发送。这涉及到维护一个基于生存时间的优先级链表，因此从链表中插入、删除的时间复杂度为 $O(\log N)$，其中 N 表示等待队列中的数据包数，一个明显的优化算法是维护一个基于每一会话的优先级链表，因为属于同一会话的数据包必须按序发送，这使得时间复杂度降为 $O(\log K)$，K 表示当前链路上所复用的会话数目。假设会话经整形后的流量特性以 $A_i(t)$ 表示，$i=1,\cdots,N$，文[14]中可知 EDF 对每一会话 i 可以提供时延保证 D_i。假设在满足下述前提下：

$$\sum_{i=1}^{N} A_i(\tau-D_i)+L_{max}\leqslant\gamma\tau,\ \tau\geqslant 0$$

当 $t<0$ 时，$A_i(t)=0$，L_{max} 表示链路上的最大传输单元。

2.6 层次性链接共享

网络能在一条链路上提供几种不同性质的共享服务，链路必须隔离来支持不同的服务级别。例如：如何在一条链路上同时有效地支持尽力传输、区分服务和保障服务。区分服务和保障服务需要 QoS 控制，为其选路，预留带宽、缓冲和处理能力等资源，并进行实时调度控制。而尽力传输服务的带宽分配可以动态改变，根据当前时刻的传输要求和网络的有效带宽进行分配。同时允许资源在空闲时被其它会话所借用。可以采用 CBQ（基于分类的队列）[15]，使用基于会话种类的队列。每一种类的会话与相应的链接带宽相联系，CBQ 其目标之一是保证不同类的基本带宽，网络多余的带宽以公平合理的方式在各种会话类中共享，在各类会话中没有必要使用同样的带宽调度算法，但是显而易见，所有类使用相同的调度算法，系统效率更高。

结束语 高速网络中端到端 QoS 调度算法的设计和实现还存在许多问题需要改进：网络系统状态和链路带宽容量变化的不确定性、传输调度算法由于其复杂性而不能很好适应高速信息传输处理时间的要求等。许多学者正不懈地致力于这方面的研究，已经有了一些基本成果并逐步被厂商应用到产品中，提高 Internet 的整体性能，但它仍然是一个未解决的问题。

参 考 文 献

1 Cruz R L. Quality of service guarantees in virtual circuit switched networks. IEEE J. Select. Areas Commun. 1995, 13(6): 1048～1056

2 Georgiadis L, et al. Efficient network QoS provisioning based on per node traffic shaping. IEEE/ACM Trans. Networking, 1996, 4(4): 482～501
3 Goyal P, Chen H, Vin H. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. In: Proc. SIGCOMM, Stanford University, CA, 1996. 157～168
4 Parekh A K, Gallager R G. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. IEEE/ACM Trans. Networking, 1994, 2(2): 137～150
5 Stiliadis D, Varma A. Latency rate servers: A general model for analysis of traffic scheduling algorithms. In: Proc. INFOCOM, San Francisco, CA, 1996. 111～119
6 Golestani S J. A self-clocked fair queueing scheme for broadband applications. In: Proc. INFOCOM. Toronto, Canada, 1994. 636～646
7 Shenker S, Partridge C, Guerin R. Specification of guaranteed quality of service. IETF RFC 2212, Sep. 1997
8 Toutain F. Decoupled generalized processor sharing: a fair queuing principle for adaptive multimedia applications. In: Proc. of the 17th Annual Joint Conf. of the IEEE Computer and Communications Societies (IEEE INFOCOM'98). San Francisco, CA, IEEE Computer Society, 1998. 291～298
9 Bennett J C R, Zhang H. WF²Q: worst-case fair weighted fair queueing. In: Proc. INFOCOM. San Francisco, CA, 1996. 120～128
10 Foster I, Kesselman C. The Grid: A Blueprint for the New Computing Infrastructure. Morgan Kaufman, San Francisco, CA, 1998
11 Shreedhar M, Varghese G. Efficient fair queuing using deficit round-robin. IEEE/ACM Trans. Networking, 1996, 4(3): 375～385
12 Katevenis M, Sidiropoulos S, Courcoubetis C. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. IEEE J. Select. Areas Commun., 1997, SAC-9(8): 1265～1279
13 Stiliadis D, Varma A. Frame-based fair queueing: A new traffic shedufing algorithm for packet-switched networks. In: Proc. SIGMETRICS'97. 1997
14 Georgiadis L, Cuerin R, Parekh A. Optimal multiplexing on a single link: Delay and buffer requirements. IEEE Trans. Infor. Theory, 1997, 43(5): 1518～1535
15 Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Networking, 1995, 1(4): 397～413

(上接第98页)

务器生成的请求对象获得用户请求信息，并调用 doGet()、doPost()或用户开发的各种方法来处理请求。具体过程如下：

```
public void service (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{ String UserID, Password
  UserID=req.getParameter("UserID");
  Password=req.getParameter("Password");
}
```

6.3 访问数据库

IBM Websphere 提供了丰富的数据库访问功能，它支持一种 JDBC 连接池(Connection Pool)技术使得不必为每次请求建立新的数据库连接，因此，通常把建立数据库连接操作安排在 Servlet 的 init()方法中，该方法仅在 Servlet 加载时执行一次。建立数据库连接后，对用户的每次请求，服务器都会在调用 Servlet 的 service()方法时把用户的请求信息传递过来，因此可以在 Servlet 的 service()中根据用户请求生成相应的 SQL 语句并执行。下面的代码段说明了 Servlet 从系统配置文件中取得连接参数，通过 JDBC 建立数据库连接的基本过程。

```
ConfigBundle = PropertyResourceBundle ) PropertyResourceBundle.getBundle("Myconfig.ini");
DbUserid = ConfigBundle.getString (" JDBCServlet.dbUserid");
DbPasswd = ConfigBundle.getString ("JDBCServlet dbPassword");
HostName = ConfigBundle.getString (" JDBCServlet.HOSTNAME");
TcpPort = ConfigBundle.getString (" JDBCServlet.TCPPORT");
Instance = ConfigBundle.getString (" JDBCServlet.INSTANCE");
url ="jdbc:oracle:thin:@"+HostName+":"+TcpPort+":"+Instance;
con=DriverManager.getConnection(url, DbUserid, DbPasswd);
```

下面的代码段说明了根据从请求对象中获得的用户请求信息生成并执行 SQL 语句、通过连接对象获得处理结果集的基本过程。

```
Qry_stmt="Select RIGHTS From USERS WHERE USERNAME="+UserID+"AND
PASSWORD="+Password+";
Statement stmt=con.createStatement();
ResultSetrs=stmt.executeQuery(Qry_stmt);
UserRight=rs.getObject(1); //得到用户权限结果
```

6.4 返回处理结果

Servlet 根据数据库处理结果，生成相应的 HTML 页面，通过响应对象返回给用户。具体过程如下：

```
res.setContentType("text/html");
    //以下3行禁止动态内容在浏览器中缓存
res.setHeader("Pragma","no-cache");
res.setHeader("Cache-Control","no-cache");
res.setDateHeader("Expires",0);
out=res.getOutputStream();
pw=new PrintWriter(out);
pw.println("(TITLE)(您的权限是:"+UserRight+")(/TITLE)");
```

结束语 本文介绍了 Java Servlet 的定义、功能、特点和开发过程，同 Microsoft 的 ASP(ActiveX Server Page)相比，Java Servlet 的最大优势是支持目前常用的各种 Web 服务器，良好的跨平台性将最大限度地保护用户的开发投资。

参 考 文 献

1 宋关福，钟耳顺，王尔琪. WebGis-基于 Internet 的地理信息系统. 中国图象图形学报，1998，3(3)：251～254
2 IBM WebSphere Application Server. Available at: http://www.software.ibm.com/webservers/